PDF Page 5 / 8

Details

Science
Volume 368, Issue 6495
Jun 2020

ARTICLE

There's plenty of room at the Top: What will drive computer performance after Moore's law?

View article page

Charles E. Leiserson, Neil C. Thompson, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lar … See all authors

Publisher American Association for the Advancement of Science
ISSN 0036-8075

the ramifications of a design choice. In the post-Moore era, it will be essential for algorithm designers and hardware architects to work together to find simple abstractions that designers can understand and that architects can implement efficiently.

Hardware architecture

Historically, computer architects used more and more transistors to make serial computations run faster, vastly increasing the complexity of processing cores, even though gains in performance suffered from diminishing returns over time (38). We argue that in the post-Moore era, architects will need to adopt the opposite strategy and focus on hardware streamlining: implementing hardware functions using fewer transistors and less silicon area.

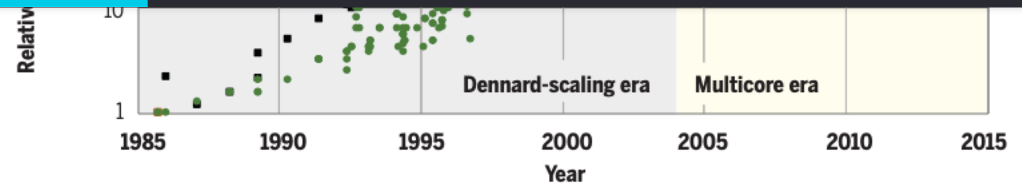

Fig. 2. SPECint (largely serial) performance, SPECint-rate (parallel) performance, and clock-frequency scaling for microprocessors from 1985 to 2015, normalized to the Intel 80386 DX microprocessor in 1985. Microprocessors and their clock frequencies were obtained from the Stanford CPU database (56). Microprocessor performance is measured in terms of scaled performance scores on the SPECint and SPECint-rate performance benchmarks obtained from (39). (See Methods for details.) Black squares identify single-core processors, for which SPECint and SPECint-rate benchmark performances are the same. Orange, blue, and red squares plot the SPECint-rate benchmark performance of various multicore processors, where orange squares identify processors with two to three cores, blue squares identify processors with four to seven cores, and red squares identify processors with eight or more cores. The gray diamonds plot the SPECint benchmark performance on multicore processors. The round green dots plot processor clock frequencies (also normalized to the Intel 80386). The gray background highlights the Dennard-scaling era (nominally up to 2004), and the white background highlights the multicore era (beyond 2004).

eventually exceeds the bandwidth of its channel to memory, so that the application is memory bound. For an application with good locality, however, as parallelism increases, the amount of off-chip memory traffic increases much more slowly, enabling all the chip's computing engines to do useful work without idling. Fortunately, many important application domains contain plenty of both locality and parallelism.

Hardware streamlining can exploit locality in other ways, especially for domain-specific processors, which we shall discuss shortly. For example, explicit data orchestration (40) exploits locality to increase the efficiency with which data are moved throughout the memory hierarchy [(41), chap. 4]. On-chip interconnects can become simpler and consume less power and area if the application using them contains locality. For example, systolic arrays (42) can perform matrix computations more efficiently using an area-efficient mesh interconnect than a general-purpose interconnect.

Although hardware will increase in capability because of streamlining, we do not think that average clock speed will increase after Moore's law ends, and it may in fact diminish slightly. Figure 2 shows that clock speed plateaued in 2005, when microprocessor design became power constrained. Before 2004, computer architects found ways to increase clock frequency without hitting hard power limits. Dennard scaling—reducing voltage as clock frequency increased—allowed processors to run faster without increasing power usage. (In practice, processor manufacturers often increased clock frequency without reducing

Processor simplification (44) replaces a complex processing core with a simpler core that requires fewer transistors. A modern core contains many expensive mechanisms to make serial instruction streams run faster, such as speculative execution [(41), section 3.6], where the hardware guesses and pursues future paths of code execution, aborting and reexecuting if the guess is wrong. If a core can be simplified to occupy, say, half as many transistors, then twice as many cores can fit on the chip. For this trade-off to be worthwhile, the workload must have enough parallelism that the additional cores are kept busy, and the two simplified cores must do more useful computing than the single complex one.

Domain specialization (11, 43, 45) may be even more important than simplification. Hardware that is customized for an application domain can be much more streamlined and use many fewer transistors, enabling applications to run tens to hundreds of times faster (46). Perhaps the best example today is the graphics-processing unit (GPU) [(41), section 4.4], which contains many parallel "lanes" with streamlined processors specialized to computer graphics. GPUs deliver much more performance on graphics computations, even though their clocks are slower, because they can exploit much more parallelism. GPU logic integrated into laptop microprocessors grew from 15 to 25% of chip area in 2010 to more than 40% by 2017 (see Methods), which shows the importance of GPU accelerators. Moreover, according to the Top 500 website, which tracks high-performance computing technology, only about 10% of supercomputers that were added

GPUs have nevertheless broadened to be handy for a variety of nongraphical tasks, such as linear algebra. Consider the matrix-multiplication problem from the Software section. An Advanced Micro Devices (AMD) FirePro S9150 GPU (48) can produce the result in only 70 ms, which is 5.4 times faster than the optimized code (version 7) and a whopping 360,000 times faster than the original Python code (version 1).

As another example of the trade-off between broadening and specialization, GPUs were crucial to the "deep-learning" revolution (49), because they were capable of training large neural networks that general-purpose processors could not train (50, 51) fast enough. But specialization has also succeeded. Google has developed a tensor-processing unit (TPU) (52) specifically designed for deep learning, embracing special-purpose processing and eschewing the broader functionality of GPUs.

During the Moore era, specialization usually yielded to broadening, because the return on investment for developing a special-purpose device had to be amortized by sales over the limited time before Moore's law produced a general-purpose processor that performs just as well. In the post-Moore era, however, we expect to see more special-purpose devices, because they will not have comparably performing general-purpose processors right around the corner to compete with. We also expect a diversity of hardware accelerators specialized for different application domains, as well as hybrid specialization, where a single device is tailored for more than one domain, such as both image processing and machine learning for self-driving vehicles (53). Cloud computing will encourage this diversity by ag-